# Estimation de tempo par fenêtrage, autocorrélation et corrélation avec des trains d’impulsions.

La méthode de détermination du tempo que nous avons mis en place est en grande partie issue de la publication *« ﻿Streamlined Tempo Estimation Based on Autocorrelation and Cross-correlation With Pulses »*, que nous appelerons à présent la publication. Cette méthode, comme d’autres méthodes de détermination du tempo, est basé sur le principe que le tempo est une périodicité de l’apparition des notes. Le signal d’onset qui correspond à cette notion d’apparition des notes est donc pseudo périodique et le tempo du morceau de musique peut être déduit de cette période.

|  |
| --- |
|  |
| Figure - Puissance d'onset des 10 premières secondes de "No Surprises" |

## Fenêtrage

La première étape consiste à fenêtrer le signal d’onset OSS en segment d’environ 5.5 secondes de long et se chevauchant de 10% (soit 0.55 secondes en commun entre deux fenêtres consécutives).

Ce fenêtrage a pour but de rendre indépendante les différentes parties du morceau. Imaginons qu’au bout de 20 secondes, le morceau contienne un long silence de plusieurs secondes. La périodicité que l’on cherche à déterminer serait complétement rompu. En fenêtrant comme décrit plus haut, on peut tenter de déterminer le tempo sur les 30 premières fenêtres obtenues (5.5 secondes + 29x0.55 seconde ≈ 20 secondes). On final on obtient M fenêtre dénotées *m*.

## Autocorrélation généralisée

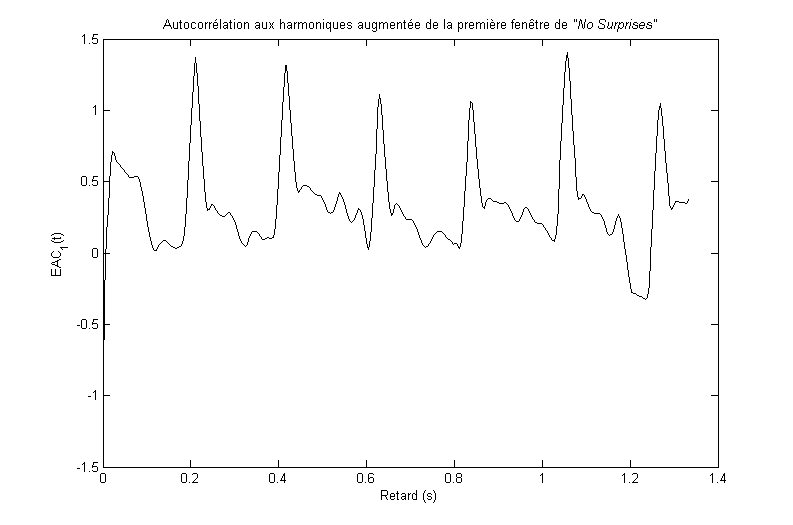
Pour chaque fenêtre obtenue, on calcul ensuite l’autocorrélation généralisée. Ce calcul se fait en passant par le domaine fréquentiel (par FFT), en compressant le spectre obtenu par un coefficient *c*, puis à revenir dans le domaine temporelle (IFFT) :

Avec c=2, on retrouve la formule de l’autocorrélation classique qui est la transformée de Fourier de la DSP du signal temporelle. Pour un *c* plus faible, les pics de l’autocorrélation sont plus étroit (plus précis). Comme dans la publication, nous utilisons un *c=0.5*.

|  |
| --- |
|  |
| *Figure 2 - Autocorrélation généralisée sur la première fenêtre de "No Surprises"* |

Renforcement des harmoniques :

Les pics que l’on observe dans l’autocorrélation représentent la périodicité du signal d’onset. On remarque cependant qu’ils sont espacés de façon régulière les uns des autres. On cherche donc à intensifier ce phénomène en multipliant l’autocorrélation obtenue avec une version d’elle-même sous-échantillonnée (ou accélérée) d’un rapport de 2, puis de 4 :



## Sélection de pics

Dans le signal EAC décrit précédemment, on sélectionne ensuite les 10 plus forts pics dans l’intervalle [0.33 s, 1.09s] correspondant aux bornes que l’on se fixe pour détecter un tempo (55 BPM et 180 BPM). On déduit du retard P de ces pics une liste de 10 tempos candidats.

## Corrélation du signal d’onset avec des trains d’impulsions

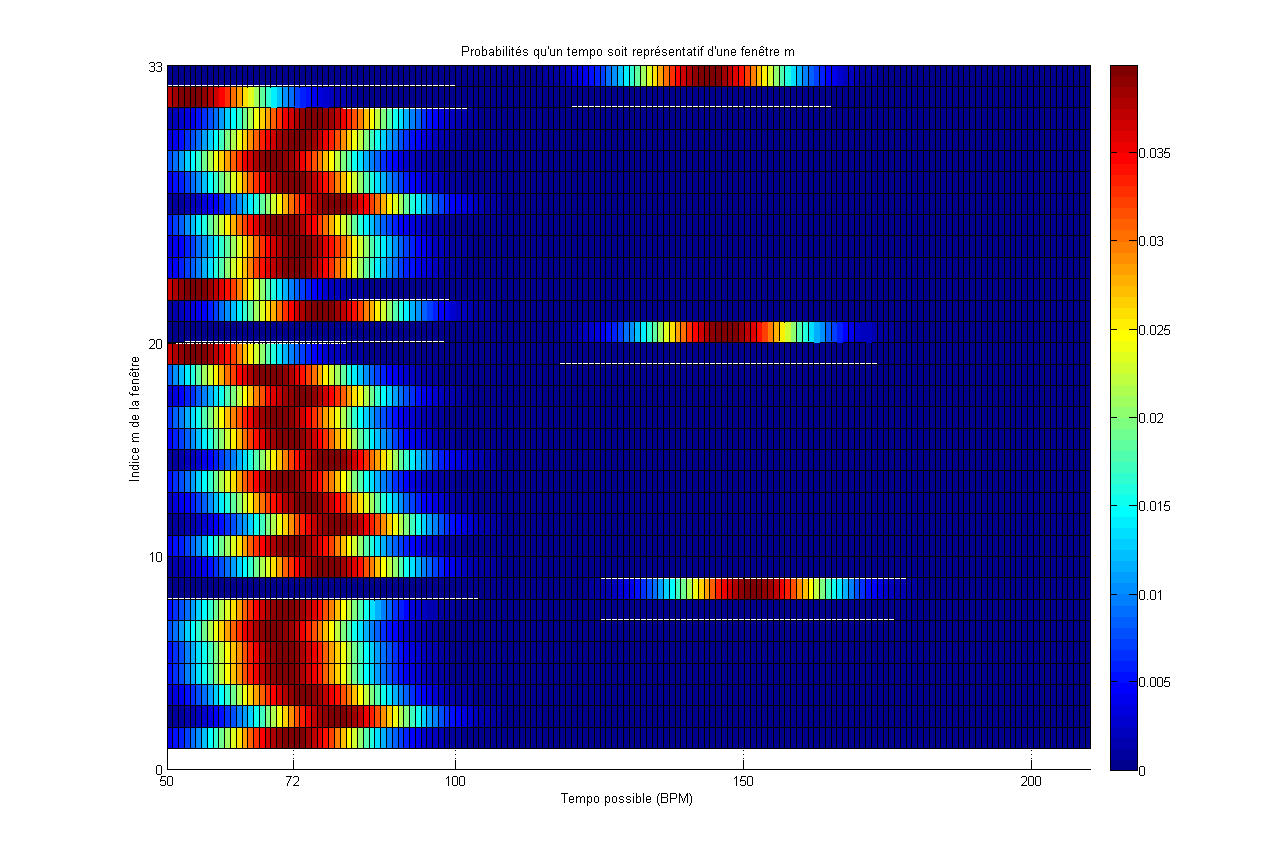
Dans le but de déterminer quel candidat est le tempo présent dans la fenêtre, on construit pour chaque candidat un train d’impulsions construit comme suit :

On place dans un train d’impulsions, 4 impulsions aux instants de retard correspondants à BxP secondes, pour B allant de 0 à 3.

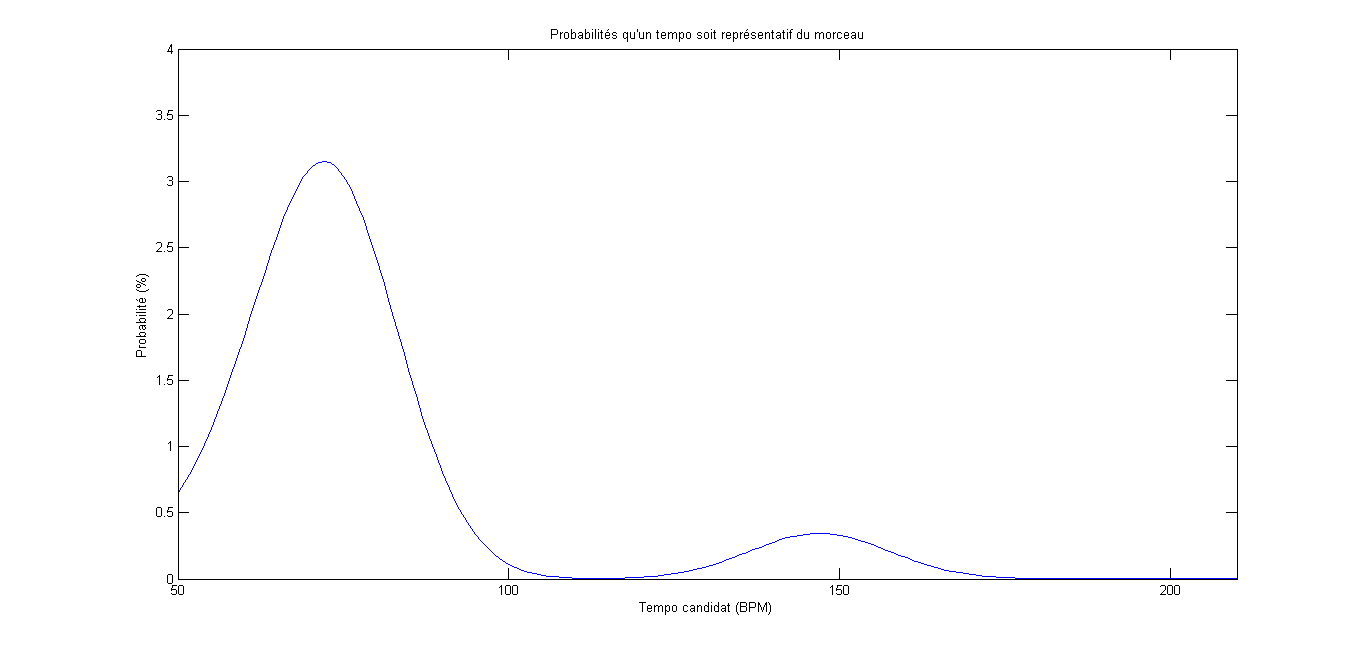
On construit 2 autres trains d’impulsions de la même façon mais avec 1.5P et 2P. On somme tous ces trains d’impulsions avec un rapport de deux en faveur du premier train.

|  |
| --- |
|  |
| Figure - Construction des trains d'impulsions |

## Détermination du meilleur candidat.



Accumulation de toutes les gaussiennes dans un seul vecteur C.



## Méthode originale décrite dans la publication

Détection des 3 pics principaux (même s’il en existe moins…). Le pic ayant la plus forte amplitude correspond au tempo candidat L. Les deux autres pics sont les deuxième et troisième meilleur candidats, L2 et L3.

Utilisation d’une Support Vector Machin (SVM) : en entrée de la SVM, il est demandé plusieurs paramètres. Le tempo sera détecté avec une certaine incertitude σ. Dans la publication, cette incertitude varie avec le tempo candidat L : . Dans notre implémentation elle est fixe à 4 BPM. Les paramètres sont les suivants (tempo = tempo réel)

1. La probabilité que le tempo soit inférieur à L :
2. La probabilité que le tempo soit supérieur à L
3. La probabilité que le tempo soit proche de L
4. La probabilité que le tempo soir proche de L/2 (rapport de ½ )
5. La probabilité que le tempo soir proche de 2L (rapport de 2 )
6. La somme de ces 3 dernières probabilités
7. Les rapports entre L et le deuxième meilleur candidat
8. Le rapport entre L et le 3ème meilleur candidat
9. Le nombre de tempo pour lequel la densité de probabilité n’est pas nulle (en mode discret)
10. Le tempo candidat lui-même, L.

Ces paramètres sont ensuite normalisés entre 0 et 1 et on applique à ces 10 paramètres, les poids de la SVM déterminés au préalable par apprentissage. Si le produit scalaire entre le vecteur 10x1 des paramètres et le vecteur 10x1 des poids de la SVM est positif, il faut doubler le tempo candidat.

Remarque : Cette méthode bien qu’elle donne des résultats convenables, nous semble assez étrange dans ses détails : la publication ne cite que les paramètres 1, 4 et 10 alors que dans l’implémentation Matlab donnée en référence apparaissent ces 10 paramètres différents.

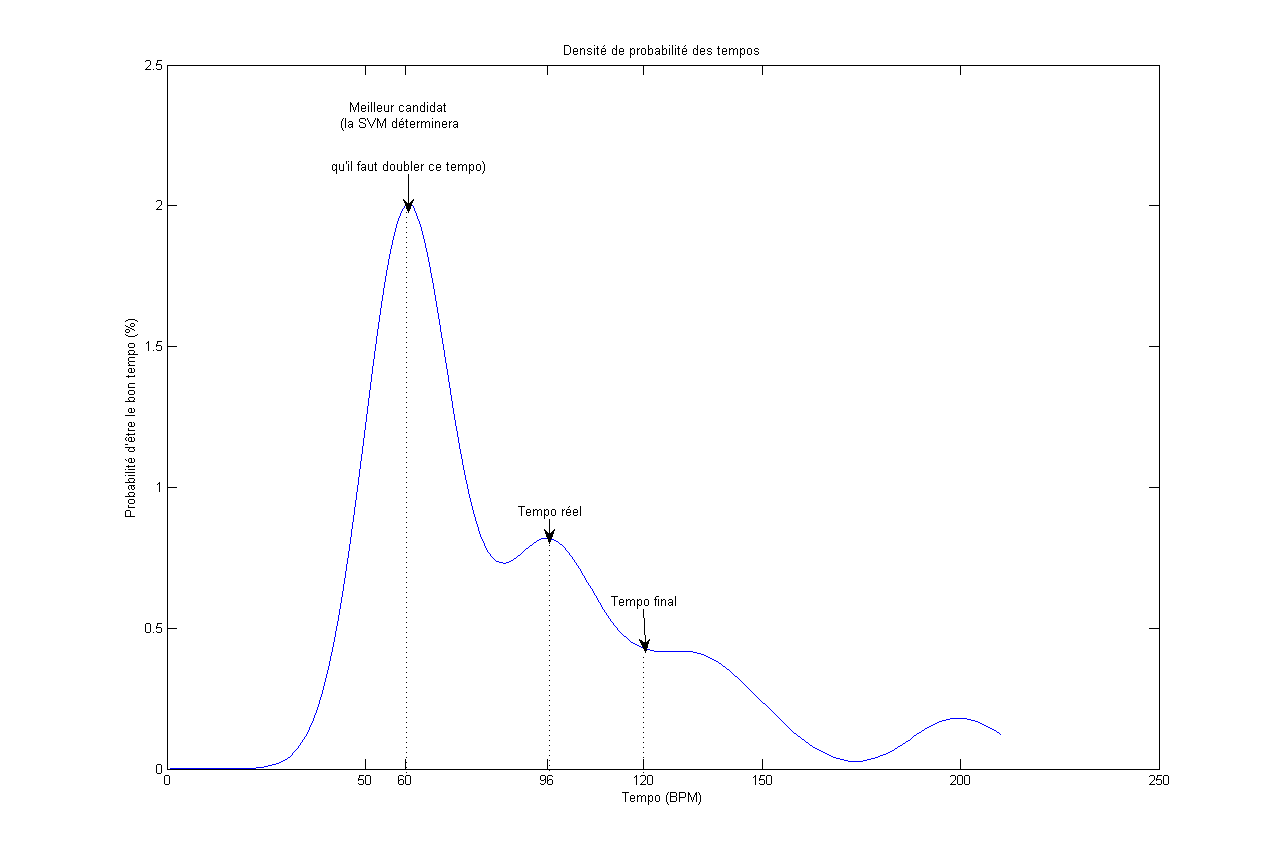
D’autre part, nous avons pu récupérer les valeurs déterminées issues de l’apprentissage mais ces valeurs donnent de mauvais résultats sur notre jeu de donnée. Il a fallu les modifier manuellement et neutralisé certains paramètres pour obtenir des résultats corrects.

Enfin, la SVM de cette algorithme ne permet de déterminé que s’il faut doubler ou non le candidat final et non s’il faut le diviser par deux ou tout simplement choisir un autre candidat.

## Amélioration proposée

### Probabilité des durées de notes

Ce qui motive notre volonté de modifier l’algorithme est notamment le fait qu’il ne peut choisir qu’entre 2 tempo possibles, le meilleur candidat et son double. Dans bien des cas, le tempo réel correspondra à la moitié du meilleur candidat, ou bien à un double/moitié d’un autre candidat :



La différence entre notre système et celui de la publication est que nous nous servons par la suite du tempo pour déterminer la durée musicale des notes (croche, noire, blanche…). Nous avons donc besoin d’un tempo fiable pour réaliser correctement cette étape. Une erreur sur le tempo à ce moment entraîne des répercussions par la suite. Dans notre exemple, une noire jouée à 90BPM sera déterminée comme une croche pointée à 120 BPM, ce qui est déjà en soi une erreur. Cette erreur aura également un impact au moment du découpage en mesure 4 :4 prévu par notre algorithme.

Or cette étape de détermination des durées est également un atout que nous pouvons utiliser. En effet, l’objectif de l’algorithme de détermination des durées est d’assigner à chaque note jouée une durée musicale ainsi qu’un coefficient de certitude que la durée a bien été déterminée. Ce coefficient correspond à la probabilité que la note jouée soit d’une certaine durée musicale, sachant le tempo estimé. Cela entraîne que si le tempo est mal estimé, cette probabilité attribuée à chaque note va être faible en moyenne, alors que si le tempo est correctement estimé, la moyenne des probabilités sera maximisée. Nous proposons donc de calculer ces probabilités pour tous les candidats.

Remarque : cette logique ne permet pas de déterminer s’il faut doubler ou diviser le tempo retenu. Il pourra toujours y avoir cette erreur d’un facteur 2 ou bien, on pourrait avoir 2 tempos candidat ex-aequo à la première place. On utilisera pour ce problème une méthode de machine learning basée sur la SVM de la publication.

On choisit donc de garder tous les maximums locaux présents dans la densité de probabilités C. Il y en a obligatoirement au moins 1 mais il peut y en avoir plusieurs, sans limite. On exécute alors la détermination des durées pour chaque tempo τ et on calcule à chaque fois la moyenne µ(τ) des probabilités p des N notes :

On considère alors que le meilleur candidat sera le τ qui maximise µ(τ). On descend également la limite supérieure de tempo acceptable de 210BPM à 180BPM.

### Fréquence des durées de notes

Un autre élément à prendre en compte est la lisibilité de la partition. Il est commun en musique d’utiliser majoritairement les durées de notes liées directement par un facteur 2 : les noires, croches et doubles-croches (les blanches et les rondes sont rares). Les durées de notes plus exotiques comme les notes pointées sont plus rares. Il faut donc faire intervenir ce paramètre dans le choix du tempo. De plus, l’écriture d’une partition étant subjective, un musicien préférera noter des croches que des noires (il choisira donc un tempo faible) alors qu’un autre préférera, pour la même partition à transcrire, utiliser des noires. Il aura alors besoin d’un tempo différent. On remarque donc que le choix du tempo dépend également de la proportion de croches (respectivement de noires, de doubles-croches etc…) que l’utilisateur voudra voir figurer dans sa partition.

|  |
| --- |
|  |
| Figure - Heart & Soul écrit avec un tempo respectivement de 120 BPM et 60 BPM |

On crée donc 4 nouveaux paramètres, la proportion de noire, croche, double-croche et autres durées de notes parmi les N notes :

|  |  |
| --- | --- |
|  |  |
|  | 1. 1- |

Où # signifie « nombre de » parmi les N notes du morceau.

Enfin, un cinquième paramètre sert à mettre en évidence une durée parmi toutes les autres. Si l’utilisateur choisi de privilégier l’utilisation des croches par exemple, on calculera de plus :

### Régression logistique

Plutôt qu’une SVM, nous choisissons d’utiliser une régression logistique (réseau de neurones sans couches cachée). Nous avons procédé à l’apprentissage des valeurs de pondérations sur les 6 paramètres présentés plus haut ainsi que sur l’unité. Cet apprentissage a été réalisé en utilisant un jeu de données différent de celui que nous avons créé. Il s’agit